Introdução ao LaTeX

Seminário LaTeX - o Livro

Geraldo Xexéo[1,2]

[1]Departamento de Ciências da Computação
[2]Programa de Engenharia de Sistemas e Computação

Março 2020

Sumário

Resumo

Esse texto é uma introdução ao LaTeXescrita em português, criada como resultado de um seminário de introdução ao LaTeXrealizado na época do Covid-19.

Ele não pretende ser a melhor introdução ao LaTeX, apenas mais uma, mas foi criada com a intenção de facilitar um pouco a vida dos meus alunos.

Capítulo 1

Introdução ao Processamento de Texto

Os computadores foram criados para fazer contas, mas, na verdade, eles manipulam apenas símbolos. Rapidamente seus usuários perceberam que podiam ser usados para manipular textos, como código, e formatá-los de alguma forma para impressão. O processamento de texto é possivelmente a área de software com o maior número de usuários, já que todo usuário de computador, alguma vez, escreveu algo e enviou para alguém, nem que seja um simples e-mail. Para isso, foram criados vários tipos de programas, que cumprem funções como as descritas na figura 1.1, criando o que pode ser chamado de uma cadeia de processamento de texto. Basicamente, o que a figura mostra é um arquivo de texto pode ser editado, visualizado, processado para impressão ou processado de formas adicionais. Cada sistema de processamento de texto faz, prioritariamente uma dessas funções[1].

1.1 Tipos de Sistema de Processamento de Texto

Dentro dessa cadeia de processamento existem vários tipos de programas. Os principais tipos são:

- Editores de Texto
- IDEs
- Processadores de Texto
- Sistemas de Autoria

[1]Observa-se que esta cadeia pode, para outros contextos, ser representada de forma mais complexa, incluindo conceito de produção, versionamento, seguração, etc.

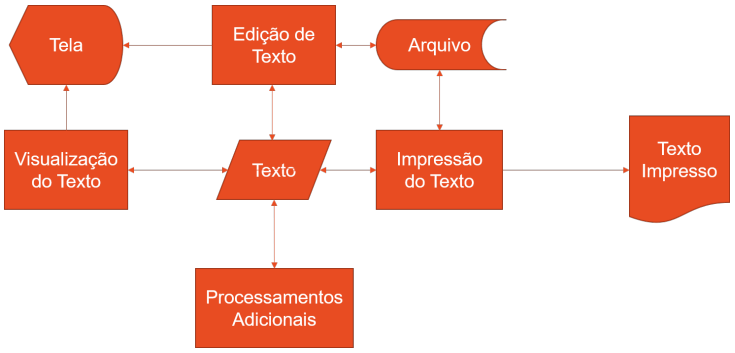

Figura 1.1: A cadeia de processamento de texto

- Sistemas de Publicação (*Desktop Publishing*)
- Sistemas de Composição
- Sistemas Colaborativos de Edição

1.1.1 Editores de Texto

Editores de texto são programas que permitem ao usuário editar arquivos que são de **texto puro**. Texto puro é um conceito que mudou. Inicialmente significava que havia um mapeamento um para um entre o que você encontrava no arquivo, uma sequência de caracteres codificados em ASCII[2]. Em ASCII, por exemplo, a letra "A" é representada pelos bits "1000001" e a "a" por "1100001". Atualmente são usadas codificações que permitem que um caracter ou símbolo seja representado por uma sequência mais longa de bits, por meio de *escape codes*, por exemplo UTF-8[3], o que significa que os editores de texto, normalmente, não representam mais perfeitamente o arquivo em disco. Os sistemas de codificação atuais são normalmente extensões do ASCII, isto é, os 128 códigos de 1 byte do ASCII original ainda são válidos. A figura 1.2 mostra a função de um editor de texto na cadeia de processamento de texto.

Editores de texto foram necessários assim que trocamos as entradas por cartão e fita, que eram editados em máquinas não conectadas ao computador, por terminais ligados diretamente aos mesmos. Os primeiros editavam linha a linha, a seguir outros exigiam que o usuário gerenciasse o *buffer*, o seja, a parte do arquivo que estava em memória. Com o tempo chegamos a versões

[2]ASCII é um padrão que associa uma letra, e outros símbolos usados em arquivos, a um byte. Seu nome significa American Standard Code for Information Interchange. Baseado no alfabeto inglês, originalmente usava apenas 128 símbolos (7 bits), não possuindo os caracteres acentuados de outras línguas.

[3]Unicode Transformation Format, que permite codificar 1.112.064 símbolos

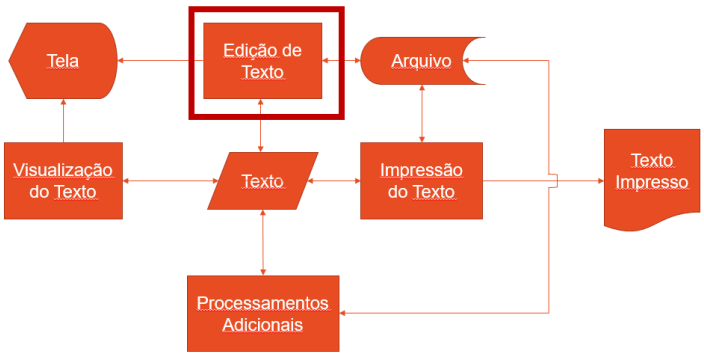

Figura 1.2: Função de um editor de texto na cadeia de processamento de texto.

semelhantes as atuais.

Atualmente editores de texto tem um conjunto complexo de funções de busca, substituição, etc.

Exemplos de editores de texto atuais são o Notepad, que vem por *default* com o Windows, o `vi` e o `vim`, Notepad++, EditPlus, TextEdit e o poderosíssimo Emacs. A figura 1.3 mostra um exemplo to `vim`.

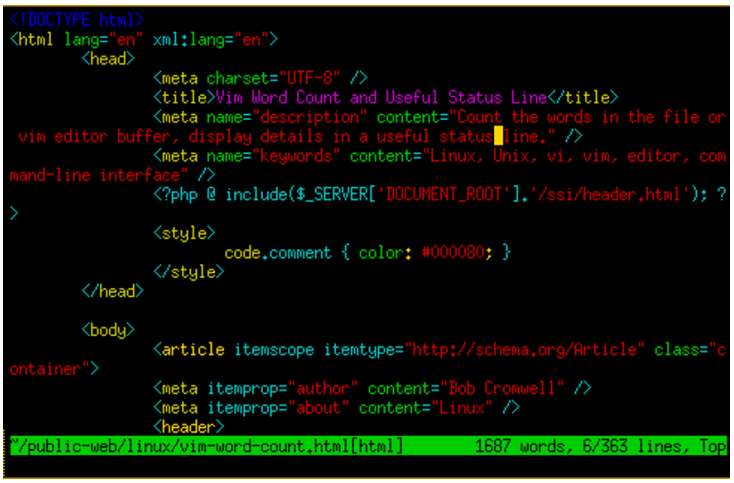

Figura 1.3: O editor de texto vim

1.1.2 IDEs

Uma IDE, ou "Integrated Development Environment", é uma extensão lógica da ideia de editor de texto criada originalmente para programadores, fornecendo serviços adicionais a edição de texto, como compilação, *debugging*,

controle de versão, normalmente por meio de interfaces com os programas que fazem isso.

Uma IDE cobre a parte de edição e processamento de texto da cadeia de processamento de texto, como visto na figura 1.4.

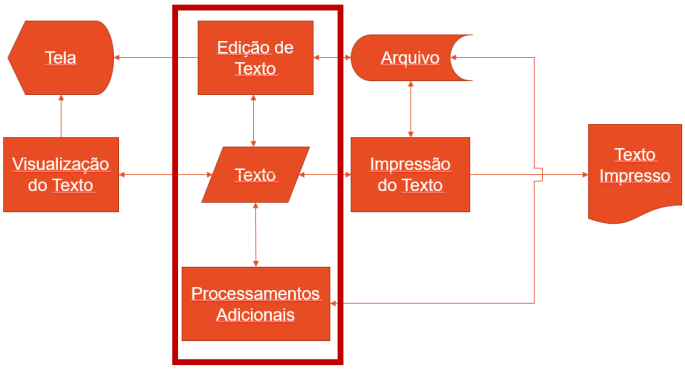

Figura 1.4: Um IDE permite editar e invocar outros processsamentos de texto

Exemplos de IDE são o Atom, o Visual Studio, o TEXStudio e o próprio Emacs. Com a evolução dos editores de texto, a fronteira entre editores e IDEs ficou indefinida, muitas vezes dependendo do uso que o usuário faz do programa. A figura 1.5 mostra um exemplo do TEXStudio.

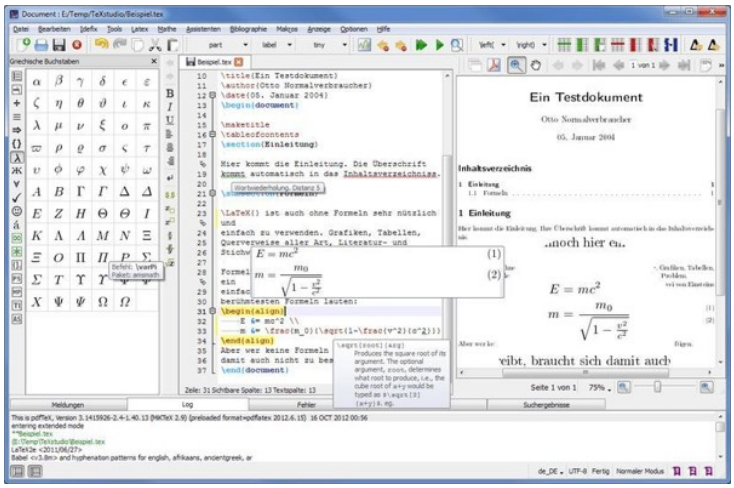

Figura 1.5: O TEX Studio

1.1.3 Sistemas de Composição (*Typesetting*)

Já que era possível editar textos, porque não imprimi-los de forma adequada? Essa ideia levou a criação de programas de tipografia, que faziam a tradução de um arquivo texto, com marcações adequadas, para um outro arquivo que fosse interpretado em uma impressora (ou outras máquinas mais sofisticadas de *typesetting* dgital).

Um sistema de tipografia cobre apenas a parte de impressão da cadeia de processamento de texto, como visto na figura 1.6.

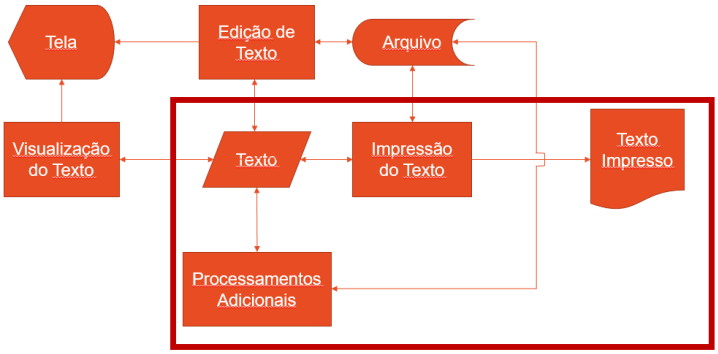

Figura 1.6: Sistemas de tipografia imprimem e fazem processamentos adicionais.

Essas marcações adequadas são como comandos, e um arquivo de texto marcado se assemelha a um programa de computador, por possuir palavras código que dão os comandos necessários. Sistemas desse tipo não possuem editores associados e são normalmente ativados por linha de comando.

Exemplos de sistemas de tipografia são o troff, o TEX e o LATEX. A figura 1.7 mostra o LATEXsendo usado em linha de comando.

Os sistemas de composição mais avançados, como o LATEX, ou sistemas baseados em SGML, como o próprio HTML, tem a característica de separar a lógica do texto, sua estruturação, da sua forma de apresentação. Isso é conseguido no LATEX por meio do uso de classes de documentos, pacotes e comandos criados pelo usuário com essa intenção. No HTML isso é conseguido por meio do uso de classes para as tags e especificações CSS que definem a aparência das tags que pertencem aquela classe. Em XML é feito por meio de um processador externo que usa uma especificação que diz como o XML deve ser apresentado, por exemplo usando XSL como linguagem para definir o estilo a ser usado na apresentação.

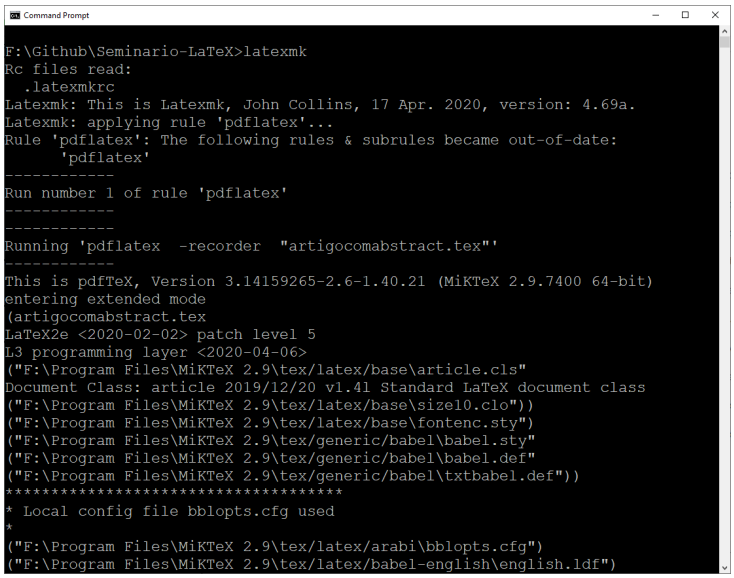

Figura 1.7: O LaTeXsendo usado

1.1.4 Processadores de Texto

Principalmente com o advento do micro-computador, ficou claro que uma das principais utilidades do computador seria permitir a criação de textos a serem publicados, logo um editor de texto deveria ser estendido para suportar funções como colocar palavras em negrito, itálico, selecionar fontes, etc.

Processadores de texto cumprem grande parte das funções na cadeia de processamento de texto, como mostra a figura 1.8.

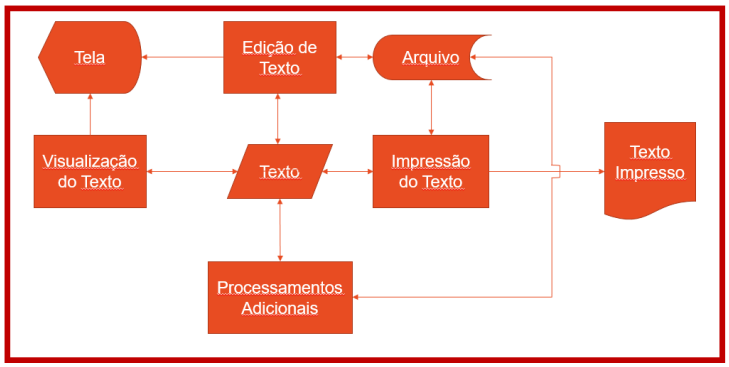

Figura 1.8: Os processadores de texto na cadeia de processamento de texto.

Com a evolução dos terminais de computadores, micro-computadores e placas de vídeo e monitores, os processadores de texto evoluíram de sistemas

que mostravam alguma coisa do que estava sendo prevista para o texto, como caracteres em bold, para sistemas que permitiam uma visualização prévia, até chegar a edição pelo conceito de WYSIWYG, lido "uiziwig", que mostra na tela quase que exatamente o que será visto na edição final, sendo as diferenças mínimas e quase imperceptíveis causadas por questões tecnológicas e da diferença entre papel e tintas e monitores.

Atualmente o Word é o processador de texto que domina amplamente o mercado, e seus principais concorrentes são sistemas de código aberto, como o LibreOffice Writer ou o Apache OpenOffice Writer. Um tela do Word é mostrada na figura 1.9.

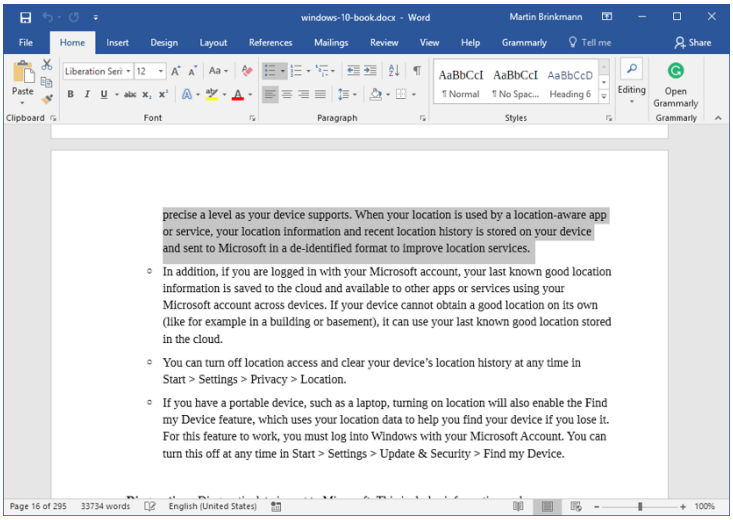

Figura 1.9: O Word é o principal processador de texto do mercado.

Uma característica avançada importante dos processadores de texto é o uso de estilos, e templates, que são basicamente coleções de estilos organizadas de modo a dar um formato coerente ao documento. Por meio do uso de estilos é possível garantir que partes do documento que tem a mesma função tenham a mesma aparência.

1.1.5 Sistemas de Autoria

Sistemas de autoria são uma evolução interessante dos processadores de texto, ou dos IDEs, voltadas para autores de livros, roteiros, etc. que não só permitem editar o texto, em formatos específicos, como também guardar informações como fichas de personagens, descrições de cena, *storyboards*, etc... Eles podem cobrir toda a cadeia de processamento de texto.

Mesmo o mercado não sinalizando isso, sistemas de autoria tem grande potencial para o futuro e para a academia, pois normalmente ao escrever algo precisamos guardar muito informação relativa ao conteúdo, e um sistema de autoria poderia ajudar a fazê-lo.

Exemplo de sistemas de autoria são o Scrivener, apresentado na figura 1.10, o Final Draft e o Celtx.

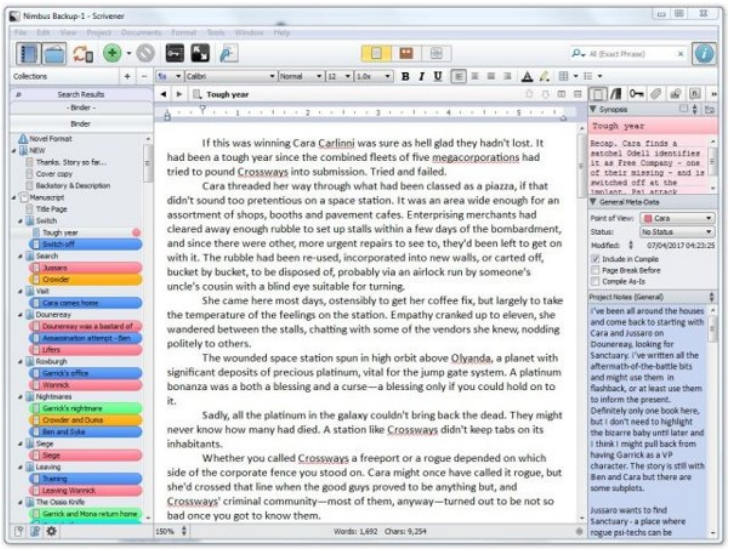

Figura 1.10: Uma tela do Scrivener

1.1.6 Sistemas de Publicação

Sistemas de publicação, ou*Desktop Publishing*, são sistemas cujo o foco é a diagramação de publicações, fornecendo uma capacidade menor de processamento de texto. São nesses sistemas que a maioria de jornais e revistas, e tamém manuais, folhetos, e outras formas de documentos impressos, são hoje preparados para a impressão.

Exemplos típicos de sistemas de publicação são o Publisher e o Framemaker, que é mostrado na figura1.11.

Sistemas de publicação são mais difíceis de usar que sistemas de processamento de texto, porém permitem fazer a diagramação de um documento de formas mais avançadas, e depois colocar os textos nos espaços dentro da diagramação. Normalmente o texto é editado em outros programas, por exemplo, em um processador de texto, que permite colocar já os formatos como negrito e itálico.

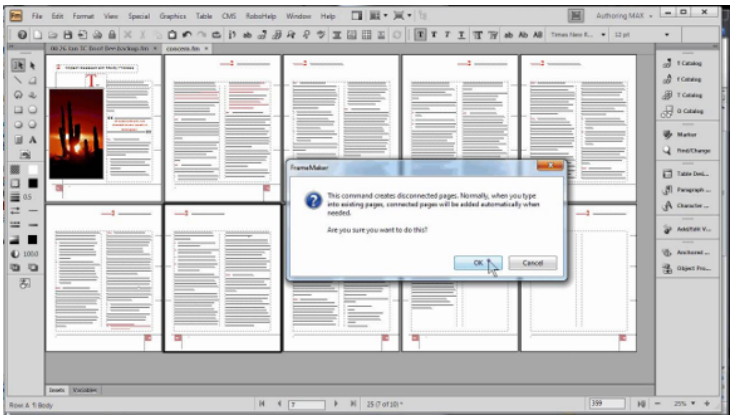

Figura 1.11: O software Framemaker

1.1.7 Sistemas Colaborativos de Edição

Esses sistemas aparecem cedo, porém se expandem principalmente com o fortalecimento da internet. Eles permitem que mais de uma pessoa edite um arquivo ao mesmo tempo. Atualmente o mais conhecido é o Google Docs, que é um processador de texto limitado em funcionalidade mas muito fácil de usar.

O ShareLateX foi um sistema colaborativo de edição voltado para o LaTeX que acabou sendo responsável por um renascimento do uso do LaTeX na academia. Acabou sendo incorporado ao concorrent Overleaf, que aparece na figura 1.12.

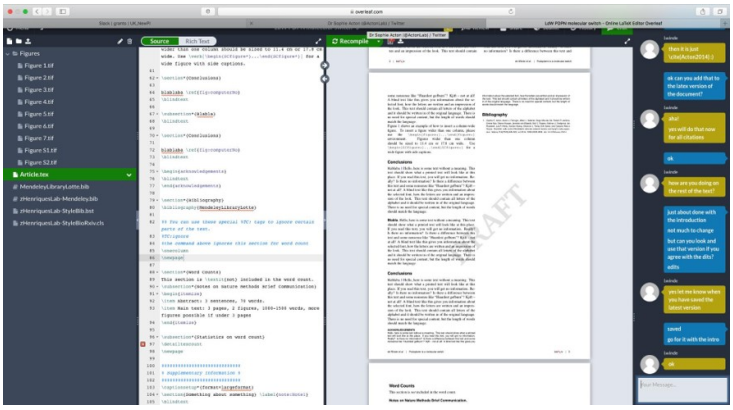

Figura 1.12: O Overleaf.

1.1.8 O que é o LaTeX

LaTeXé um sistema de composição, ou *typesetting*, baseado no TeXe apoiado com outros programas, como o biber. que permite usar arquivos de texto marcados para criar arquivos a serem impressos, possivelmente no formato PDF, seguindo regras de composição e usando fontes detalhadamente criadas para reproduzir a qualidade de fontes utilizada na composição manual, incluindo especialmente as ligaduras, que são caracteres especiais que representam dois ou mais caracteres de forma visualmente mais elegantes, como mostradas na figura 1.13.

$$\begin{array}{lcllcl} fi & \longrightarrow & \textit{fi} & AE & \longrightarrow & Æ \\ ff & \longrightarrow & \textit{ff} & ae & \longrightarrow & æ \\ ffi & \longrightarrow & \textit{ffi} & OE & \longrightarrow & Œ \\ fl & \longrightarrow & \textit{fl} & oe & \longrightarrow & œ \\ ij & \longrightarrow & \ddot{y} & LATEX & \longrightarrow & \LaTeX \end{array}$$

Figura 1.13: Exemplos de ligaturas de fontes do TeX.

1.2 Tipos de Linguagens para Arquivos de Texto

Arquivos de texto podem ser guardados de várias formas. No limite, podemos fotografar uma página de texto e guardar a imagem, mas geralmente queremos um arquivo que possa ter seu texto manipulado.

Normalmente, na cadeia de processamento de texto, o arquivo é lido e colocado em memória em um formato mais adequado, e depois, quando salvo, é "rearrumado" de alguma forma, um formato de arquivo texto. Por exemplo, ao salvar um arquivo Word você pode escolher o seu formato nativo (.docx) ou vários outros formatos alternativos, como RTF[4], ou mesmo um arquivo texto, nesse caso perdendo toda a formatação.

Os formatos de arquivo podem ser divididos em:

- Linguagens de Impressão/Visualização
  - PostScript, DVI, PDF
- Linguagens Intermediárias (de impressão)
  - .dvi

[4]Rich Text Format

- Linguagens de Marcação
    - SGML, HTML, TEX LATEX Markdown, RPF, fods

O fato de uma linguagem de marcação ser legível por humanos não quer dizer que seja facilmente legível, principalmente quando geradas por máquinas. A figura 1.14 mostra um exemplo de arquivo Postscript.

Figura 1.14: Exemplo de um arquivo PostScript

1.2.1 Linguagens de Marcação

Um linguagem de marcação é caracterizada por um conjunto de códigos que é aplicado sobre o texto, ou sobre qualquer forma de dados, com o fim de adicionar informações específicas sobre esse texto, a cada trecho específico, como por exemplo a forma de exibi-lo graficamente.

Linguagens de marcação são uma forma simples de indicar como um texto deve ser impresso, e são usadas desde o início da digitalização do processo de composição e impressão.

As linguagens de marcação padronizadas permitem que programas diversos cumpram a mesma função. Elas podem ser divididas em:

- Procedurais
    - troff, TEX , LATEX, Postcript
- De apresentação
    - Wikis, Markdown
- Descritivas
    - SGML, HTML, XML

Linguagens de marcação procedurais normalmente incluem instruções de composição, ou seja, elas mantém unificadas a estrutura e a apresentação,

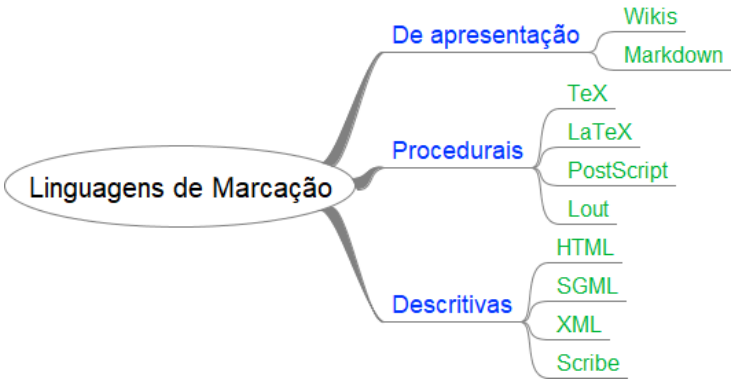

Figura 1.15: Mapa mental das linguagens de marcação.

enquanto linguagens de apresentação não se preocupam com nada além da apresentação imediata. Já linguagens descritivas, introduzias por Scribe, separam a estrutura da apresentação. LaTeX, apesar de ainda conter instruções de composição, foi fortemente influenciada por essa ideia, e o usuário final praticamente só usa instruções que falam sobre a estrutura do documento. Um mapa mental das linguagens pode ser visto na figura 1.15(Adams 2007).

Linguagens de marcação normalmente são criadas para serem simultaneamente compreensíveis para o ser humano e tratáveis por um programa de computador.

Um exemplo de código LaTeX e seu resultado é mostrado na figura 1.16.

```
1  \textbf{Um exemplo}
```

Um exemplo

Figura 1.16: Exemplo de código LaTeX

1.2.2 Linguagens de Impressão

Linguagens de impressão tem como finalidade servir apenas para o processamento por um mecanismo de impressão ou um software específico de visualização. Nesse caso, não há preocupação com a comprensão do formato por seres humanos.

Exemplos de linguagens de impressão são o formato PDF, que é um Postscript empacotado de forma a descrever documentos compostos de páginas.

TEXtrouxe o conceito de uma linguagem de impressão intermediária, por meio do formato DVI, que torna o arquivo gerado pelo TEX independente do processamento final, o que permite que cada fabricante, ou interessado em geral, faça um programa próprio de conversão de DIV para qualquer formato de impressão. Logo, documentos DVI podem ser transformados em PDF, PS ou outro formato por software especiais, mas atualmente as implementações de LATEXtornam isso transparente.

1.3 Sistemas Mais Usados na Computação

Três sistemas são hoje muito usados na Computação:

- **Word**
  - Um software WYSIWYG
  - Líder do mercado
  - Superpoderoso
  - Difícil de usar para trabalho colaborativo
- **Google Docs**
  - Quase WYSIWYG
  - Sucesso entre os jovens
  - Pouca capacidade de plena expressão gráfica
  - Ótimo para trabalho colaborativo
- LATEX
  - Melhor imagem de texto, mas no detalhe
  - Ótimo para Matemática
  - Difícil de usar
  - Empoderado pelo Overleaf
  - Renovado com os sistemas colaborativos

Desses sistemas, claramente o Google Doc ainda não serve para gerar documentos com formatação de qualidade, ou documentos destinados a artigos de revistas ou livros. Então ficaremos, na próxima seção, apenas com a discussão dos mitos e fatos que aparecem na discussão do uso de Word vs LATEX.

Chamamos a atenção que não consideramos viável o uso para objetivos acadêmicos dos sistemas abertos, como Open Office ou LibreOffice, por possuírem problemas de compatibilidade dos resultados de visualização entre versões dos próprios e com formatos disponíveis para o Word[5].

[5]Será isso um mito? Não é o que parece pela experiência do autor

1.4 Mitos e Fatos de Word vs LaTeX

1.4.1 LaTeX é mais produtivo

A suposição seria que, como você não se preocupa com o resultado final, já que ele é controlado pela classe de documento e estilos. Isso é falso, e já foi comprovado em pesquisas científicas que a produtividade é igual ou menor.

1.4.2 A qualidade de saída do LaTeX é muito melhor

Verdade, mas possivelmente só para quem entende o que está vendo. A qualidade da saída proporcionada pelo TeX depende de algum conhecimento do que é uma boa fonte, uma boa distribuição de texto. É possível que um leitor tenha alguma sensação de beleza ou conforto superior quando vê um texto gerado pelo TeX , mas é possível também que não saiba diferenciar, pois são detalhes.

1.4.3 Word não trabalha bem com fórmulas matemáticas

Falso. A nova maneira de trabalhar com fórmulas do Word, que já está funcionando há algum tempo, não só é muito boa como permite escrever as fórmulas com a mesma sintaxe do que o LaTeX.

Ainda há a questão da beleza das fórmulas, mas a diferença também é pequena hoje em dia.

1.4.4 Você perde muito tempo com besteira no LaTeX

Verdadeiro. Como LaTeX é uma linguagem compilada, você pode perder muito tempo com erros espúrios. Além disso, a capacidade de detectar o erro no momento certo é fraca no LaTeX, e coisas como parenteses faltando podem causar mensagens de erro muito longe. Porém, com o uso, isso tende a diminuir.

1.4.5 Não consigo colocar a imagem onde quero no Word

Falso. Sinceramente, nem sei de onde vem esse mito. Você pode colocar a imagem onde quiser no Word. O mesmo, com o LaTeX, já é mais díficil.

1.4.6 Em LaTeX gerencio melhor as bibliografias

Falso. Na verdade, LaTeX não gerencia a bibliografia, ele implica no usa outro sistema, o BibTeX, por exemplo, e outro processador, como o biber. O mesmo pode ser feito com o Word, usando sistemas como Zotero ou Paper.

1.4.7 Em LaTeX consigo controlar versões

Falso, parcialmente. Este autor é um grande fã do uso de gerência de versões, sendo um usuário do Git e do GitHub. Realmente é fácil fazê-la com o LaTeX, porque todos os arquivos são textuais e plenamente controlados no controle de versões, porém um efeito quase similar para a visualização de diferenças pode ser obtido com a instalação do software `pandoc` e uma pequena configuração do Git. Outras funcionalidades, como o *merge* de versões com conflito não são possíveis.

1.4.8 Word é mais fácil de aprender

Verdade, porém o usuário médio não usa corretamente as funções que tornam o software Word um programa fantástico, como o uso de templates e estilos.

1.4.9 Com o Word, basta ele

Falso. Para fazer tudo que você faz com o LaTeX, você precisa de outros programas para o Word também.

1.4.10 Word tem problemas com arquivos grandes

Verdade, para arquivos muito grandes, o Word realmente às vezes se perde e causa bugs, quase impossíveis de consertar, com arquivos muito grandes. A alternativa de usar arquivos *master* que incluem outros arquivos, o que se faz com facilidade no LaTeX, está quebrada há várias versões e não há perspectiva de ser corrigida[6].

1.5 Recomendações

Essas são as minhas recomendações de quando usar que sistema:

- **Word**

[6]Sim, isso é algo bizarro de se saber.

  - Ótima opção para um autor e um revisor
  - Bom para arquivos pequenos e grandes, não use para arquivos muito grandes.
  - Separe o texto em capítulos, e só junte na hora de imprimir.
  - Cuidado com o backup, faça sempre
  - Se possível, use o controle de versões
  - Aprenda a usar estilos, principalmente de parágrafos, e templates
  - Não separe parágrafos com linha em branco e não comece parágrafos com tab, use estilos para isso.
- **Google Docs**
  - Ótimo para colaborações entre várias pessoas escrevendo ao mesmo tempo
  - Sem o controle de quem fez o que, o que é possível com controle de versão (opção *Blame* no GitHub)
  - Funciona bem com arquivos pequenos, não foi testado com arquivos grandes ou enormes
- **LaTeX**
  - Se a editora fornece o formato (.sty) é a melhor opção
  - Para exames, dissertações e teses da Coppe é a melhor opção
    - Use o formato CoppeTeX disponíveis em: `https://github.com/COPPE-UFRJ/CoppeTeX`
    - basta baixar todo o diretório dist
  - Use o JabRef ou o Zotero com Better Bibtex for Zotero
  - LaTeX no Overleaft
    - Bom para artigos
    - Pode não conseguir compilar uma tese, pois tem limite de tempo.
    - Ligue as opções GitHub e Dropbox, e faça backup
  - LaTeX em casa
    - Use o Git+Nuvem (GitHub, GitLab, etc.)
    - Mantenha a instalação atualizada
    - Faça backup

Bibliografia

Adams, Nico (mar. de 2007). *Polymer Informatics and The Semantic Web The Solution, Part 1: Adding Structure*. URL: https://semanticscience.wordpress.com/2007/03/30/polymer-informatics-and-the-semantic-web-the-solution-part-1-adding-structure/ (acesso em 03/2007).